Watermarking 3D Objects for Verification

Boon-Lock Yeo and Minerva M. Yeung
Intel Corporation

The watermarking technique proposed here enables the verification of 3D polygonal models for detecting unauthorized alterations.

We have entered an era where inexpensive, readily available tools and equipment can replicate, manipulate, and distribute digital multimedia materials with ease. Most often, replication of digital content can produce perfect copies of the original, manipulation and editing of digital copies can deceive even the most professional eyes, and mass distribution can take place in a matter of seconds in electronic forms. Misappropriation of digital assets greatly concerns content owners and creators, especially when the content is made available through the Internet. Without the assurance of proper protection against lost revenues, the owners of digital content will be reluctant to make these assets available.

Many view digital watermarking as a potential solution for copyright protection of valuable digital materials like CD-quality audio, publication-quality images, and digital video. The field of digital watermarking is relatively new, and many of its terms have not been well defined. Among the different media types, watermarking of 2D still images is comparatively better studied. Inherently, digital watermarking of 3D objects remains a difficult problem. We believe a complete solution will not employ a generic algorithm to watermark objects for a variety of applications. Instead, different classes of applications will impose different requirements and call for different watermarking techniques, ranging from slight modification of existing algorithms to completely orthogonal methods.

Nonetheless, we intend to introduce and investigate the fundamental similarities and differences of watermarking 3D graphic models compared to 2D images, and propose some solutions to address a class of applications of digital watermarking-the verification of 3D polygonal models. To our knowledge, watermarking of 3D objects for verification purposes has not been addressed in any published literature. The proposed scheme, in its present form, is not intended for use in applications that require robust watermarks. One recent work in 3D data hiding addressed applications requiring robust means of hiding data.[1]

We will first introduce digital watermarking, discuss its goals and application domains, and explain the different categories of watermarks. We believe the goals and applications will remain fairly similar for 2D images and 3D models.

Digital watermarking

We define watermarking as a process that embeds data into an object-a watermark. The object can be an image, an audio clip, a video clip, or a 3D model. Some papers discuss watermarking other forms of multimedia data such as sound clips.[2,3] Since our research focuses on visual data, we say visible and invisible when in a wider sense we mean perceptible and imperceptible. We further define visible watermarking as embedding data intentionally perceptible to a human observer, while invisible watermarking embeds data not perceptible, but extractable by a computer program.

In the image domain, a variety of invisible watermarking schemes have been reported in recent years,[2-8] along with commercial systems like Digimarc's. Two categories broadly classify these techniques: spatial-domain and transform-domain based. The earlier watermarking techniques reported were spatial in nature, the simplest being the ones that modified the least-significant-bits (LSB) of an image's pixel data? Other literature proposed improvement and variants of this technique.[2,4]

Transform-domain-based techniques can be employed with common image transforms like discrete cosine transforms (DCT), wavelets, Fourier transforms (such as the FFT), and Hadamard transforms. Zhao and Koch reported one of the earlier transform-domain-based techniques tailored to JPEG-lossy image compression.[8] Other techniques include the work of Swanson, Zhu, and Tewfik.[6] Cox et al. proposed a more robust techniaue based on spread-spectrum principles.[3]

Form SF298 Citation Data

Report Date *("DD MON YYYY")* 01011999	**Report Type** N/A	**Dates Covered (from... to)** *("DD MON YYYY")*

Title and Subtitle Watermarking 3D Objects for Vevification	**Contract or Grant Number**
	Program Element Number
Authors	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Microcomputer Research Labs Intel Corp., SC12-303 2200 Mission College Blvd. Clara, CA 95052-8119	**Performing Organization Number(s)**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Monitoring Agency Acronym**
	Monitoring Agency Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes	
Abstract	
Subject Terms	
Document Classification unclassified	**Classification of SF298** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** unlimited
Number of Pages 11	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1/1/99	Article

4. TITLE AND SUBTITLE
Watermarking 3D Objects for Verification

5. FUNDING NUMBERS

6. AUTHOR(S)
Boon-Lock Yeo and Minveva M. Yeung

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

IATAC
Information Assurance Technology Analysis
Center
3190 Fairview Park Drive
Falls Church VA 22042

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Technical Information Center
DTIC-IA
8725 John J. Kingman Rd, Suite 944
Ft. Belvoir, VA 22060

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE
A

13. ABSTRACT ***(Maximum 200 Words)***

The watermarking techinque proposed in this article enables the verification of 3D polygonal models for detecting unauthorized alterations.

14. SUBJECT TERMS
Watermarking,

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	UNCLASSIFIED	UNCLASSIFIED	None

They embed a set of independent and identically distributed samples drawn from a Gaussian distribution into the perceptually most significant frequency components of the data. These papers have shown the techniques to be somewhat robust against various image-processing operations, and a few techniques can even recover the watermarks in some test images after printing and rescanning.

Depending on the end applications, watermarking techniques can be further classified into fragile and robust watermarking. In fragile watermarking the embedded watermark will change or disappear if a watermarked object is altered. In other words, the watermark can be used for verification of an object. Applications include providing trustworthy images captured with a digital camera for inclusion into news articles and trustworthy objects that will be delivered to remote clients. An invisible watermark is embedded at capture or creation time; its presence at the time of publication or upon receipt is intended to indicate that the image or object has not been altered. In another application, objects like VRML[9] models or human-fingerprint images have been delivered or scanned and stored in a digital archive. In this case, the content owner may want to detect any unauthorized alteration without comparing the objects to the original models or scans, or sending separate signature files for authentication via digital signature schemes.

In robust watermarking, the embedded watermark will persist even in the face of removal. Such removal can be unintentional transformations like filtering, cropping, translation, rotation, resizing, and lossy compression; or intentional by processing the objects to remove the watermark. Intentional attack can include any combinations of the aforementioned transformations. The embedded robust watermarks can serve as evidence of ownership if the owner's label is detected in a suspected copy. The watermarks can also serve as identification in the fingerprinting application. Here the seller of an object imprints an invisible label to indicate to whom the object is sold. If the seller later finds a copy of the object published without royalty payment, the watermarks can help identify the misappropriator. Furthermore, the embedded watermarks may serve as a form of copy protection-the detection of these indicators can trigger protection or royalty collection mechanisms like the current copy protection standard. Such a standard would incorporate digital watermarking under consideration in the Digital Versatile Disc (DVD) standardization committee, plus some commercial systems advertising the capabilities of "web-crawling" copyrighted images.

The end applications of the embedded watermarks impose specific criteria and requirements on the corresponding watermarking schemes that will give rise to the desired properties of the watermarks. This is why we avoid the term "data hiding" or "data embedding" and prefer the term watermarking. Data hiding refers to the process of hiding data in an object or media type in general and does not distinguish the end applications, nor does it describe adequately the desired properties of a particular class of watermarks.

Related work

Watermarking research is relatively new in many fields. In the previous section, we briefly introduced some published image watermarking techniques, mostly from signal-processing perspectives. A majority of existing techniques focus on embedding data into still image and video, some of which has shown a certain degree of robustness against JPEG compression, filtering, limited cropping, and rotation. Interestingly, many have equated "robust" data hiding to adequate copy right protection without a clear understanding of the limitations. Such arguments often understate the requirements and overstate the goals and benefits of the techniques developed. For example, counterfeit-watermarking schemes can be developed to allow multiple claims of rightful ownerships on an image, despite the image being watermarked. The attempt to remove the watermark is not even necessary. (A recent study on the classes of watermark attacks appears elsewhere.")

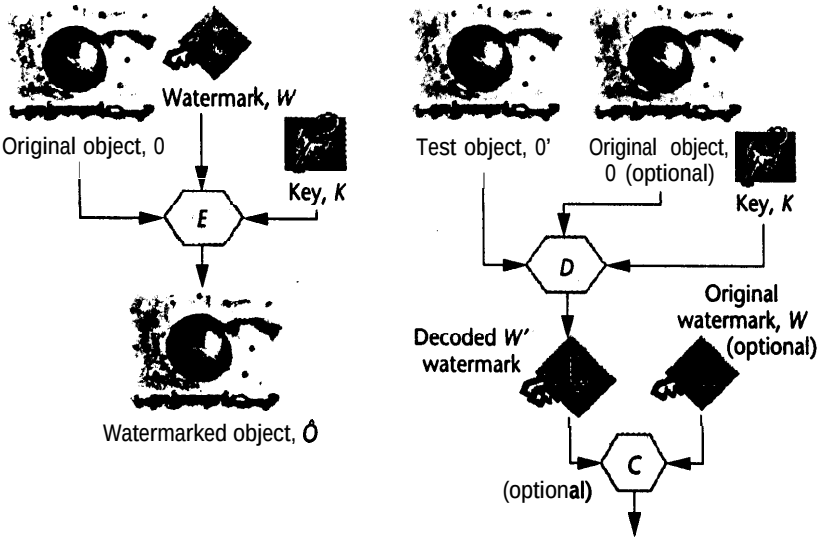
Watermarking techniques not designed carefully for 3D objects can easily fall prey to trials and tribulations similar to those experienced by image watermarking. In this article, we provide the background and general formulations of watermarking 3D objects, and address the watermarking of 3D models for verification applications. We focus on fragile 3D watermarking, which has a clearly defined set of requirements and a clear goal of detecting and locating unauthorized alterations. We do not intend to address the entire spectrum of issues in 3D watermarking and present solutions for every application.

In robust watermarking, the embedded watermark will persist even in the face of removal.

Embedding data in 3D models

To our knowledge, only one published research paper addresses the watermarking of 3D objects. Data hiding in 3D polygonal models was recently studied to address copyright protection concerns in VRML models.' The embedding algorithms operate on 3D polygonal models of geometry by modifying either the vertex coordinates, the connectivity, or both. The schemes reported do not need the reference original models to extract the embedded data. In addition to invisible data embedding, the literature also discusses a method to embed patterns into a model for visible water-marking with a focus primarily on embedding data. The embedding methods reported incorporated certain robustness criteria against some transformations like resection and local deformation by using repeated strings of data bits. A randomization of coordinates-a more general class of geometric transformation, remeshing, and so on—can destroy the embedded data.

In the reported work, the techniques provide little security to combat attempts at removal. In fact, anyone who knows the data embedding and extraction algorithms can extract the embedded data bits, and subsequently remove them. An optional stego-key described was not integrated into the watermark embedding and extraction processes; rather, it is a key for encrypting a



1 **Encoding, decoding, and comparing embedded watermarks in an object.**

readable message into a sequence of bits for embedding purposes-even if someone recovers the bit sequence, they cannot read it. But an attacker does not need to know the message before he can remove the entire bit sequence. Since the authors addressed robust watermarking applications, the lack of protection and secure decoding may eventually defeat the entire purpose of watermarking the models.

We believe that any watermarking scheme should incorporate some form of secure means in the watermark embedding and extraction processes. (Note that the degree of security may differ from the stricter requirements in data encryption and decryption.) This belief influences our formulations, described in later sections.

In robust watermarking research, many issues have not been well defined, let alone resolved. These include, but are not limited to, the spectrum of possible attacks and transformations, performance measurements, and other issues. Ohbuchi, Masuda, and Aono's study' provides a beginning vertex in the quest for better and more robust water-marking techniques. Understandably, these issues warrant further detailed study in the future.

Image verification with digital watermarks

In the image domain, Yeung and Mintzer described a technique[7] for quickly verifying that the content of an image has not changed since the image was watermarked via the use of an invisible watermark embedded into the image pixel values. This "fragile" watermarking technique consists of a watermarking process that stamps a watermark pattern invisibly onto a source image using a verification key, plus a watermark extraction process that decodes the embedded watermark from a watermarked (or stamped) image, based on the verification key. The watermark pattern is designed to allow quick visual inspection.

In the watermarking process, a binary map of a watermark image is embedded into the source image. The processing applies a watermark extraction function to the selected pixel, tests the extracted watermark value to determine if it equals the desired watermarkvalue, and, if necessary, modifies the pixel value to produce the desired watermark bit value. The modification is coupled with a modified error-diffusion process, which extends the error-diffusion techniques in image half-toning to produce small and random changes. This smoothly spreads out errors produced locally. The watermark extraction function, $WX(\,)$, is computed from a given verification key and can be implemented in the form of binary lookup-tables. In the image verification stage, a watermark image is extracted from a test image (presumably watermarked) by first computing the watermark extraction function from the verification key, then applying the function to every pixel to obtain the extracted watermark pixel. The extracted watermark image lets users visually detect unauthorized alterations made to the image (post-watermarked) and verify or authenticate the content.

The goals and the principles of 2D-image and 3D-object verification have significant similarities. However, the method employed in the image domain cannot be directly mapped to the 3D objects.

A generalized model for watermarking 3D objects

For better understanding, we present the definitions and formulations of watermarking 2D and 3D objects, and discuss the 3D features and verification process further. Also note that the formulations presented here are the generalized forms for both robust and fragile watermarking.

Definitions and formulations

We have introduced the terminology and different classes of watermarking schemes. We next present the generalized formulations of' invisible watermarking schemes. We define in general terms the process of watermarking (watermark insertion/embedding) an object and the process of watermark extraction. Figure 1 illustrates both the watermarking process, by which a watermark is inserted into an object, and the extraction process, by which a watermark is recovered and then compared to the inserted watermark.

Here we denote an object, 0; a watermark, W, comprising a sequence of owner-supplied data bits $W = \{w_1, w_2, \ldots\}$; and the watermarked object, 0. In addition, we also define a key, K, which is a sequence of bits that helps define the specific mapping function for additional security in watermark insertion and extraction. E is an encoder function if it takes an object, 0, and a watermark, W, incorporates the mapping supplied by a key, K, and generates a new object called the watermarked object, $\hat{O}$, for example, $E_K(O, W) = \hat{O}$. Note that we do not exclude the possibility that the watermark, W, is dependent on the object, 0. In such cases, the encoding process described above still holds. Also note that in some watermarking schemes the watermark, W, can actually be the key, K, or it can include the information provided in K. In other words, $E_K(O,) = \hat{O}$.

A decoder function D takes an object, 0' (0' can be a

watermarked or unwatermarked object, and possibly corrupted), and recovers a watermark, W, or evidence of the original watermark's presence, W, from the object. If available, a decoding key, K, can help define the specific mapping of the decoding function. In this process, an additional object, 0, can also be included, often the original (and unwatermarked) version of 0'. This happens because some decoding schemes may use the original or reference objects in the water-marking process to provide extra robustness against intentional and unintentional corruption of pixel values. More formally, if the decoding scheme involves a reference 0, D_K (0', 0) = $P(W)$, where P is a function indicating the presence of a watermark, W, in 0'. We call this type of watermarking scheme a "private" watermarking scheme. When $P(W)$ = W, the decoding simply returns the extracted watermark W. P may also be of the form $P()$ = $Evid()$ that returns a scalar value indicating the evidence of the presence of Win 0'. Again, in some schemes the watermark itself may suffice as the key for extraction. In such cases the information encoded is already anticipated in the decoding process, like checking whether a presumably watermarked object contains a particular data sequence supplied by its owner.

If the decoding does not need 0 in the watermark extraction, we can write a general decoding function as D_K (0') = $P(W)$. This type of watermarking scheme is called a "public" watermarking scheme. Note that in some water-marking schemes, the decoding function, D, may depend on the specific watermark embedded in the encoding process.

When $P(W)$ = W, the extracted watermark, W, is then compared to the reference watermark, W, by a comparator function, C_δ, and a binary output decision is generated indicating a match or otherwise:

$$C_\delta(W', W) = \begin{cases} 1 & c \geq \delta \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Here, c is the correlation of the two watermarks. Without loss of generality, a watermarking scheme can be treated as a three-tuple (E, D, C_δ), such that $D(E(O,W))$ = W for any object, 0, and any allowable watermark, W.

In robust watermarking, the encoding and decoding functions have to be designed such that, given the test object, 0', is a derived or transformed object from the watermarked object, $\hat{O}$, that is, $O'=T(\hat{O})$ for some transformation T: (1) if the decoding returns a value $P(W)$ = $Evid()$ indicating the presence of the watermark W in the test object 0', then the value has to be sufficiently large; or (2) if $P(W)$ = W, then C(W, W) = 1 for a sufficiently large c.

In fragile water-marking the encoding and decoding functions have to be designed such that, given the test object O' deviates from the watermarked object 0, that is, $O' \neq \hat{O}$: (1) if the decoding returns a value $P(W) = Evid()$ indicating the presence of the watermark W in the test object 0', then the value has to be sufficientlysmall; or (2) if $P(W)$ = W, then C(W, W) = 0 for a sufficiently large c.

Both robust and fragile watermarking schemes can be further classified into public and private schemes according to whether a scheme requires using the reference original 0 in the decoding. In practice, since fragile watermarks are used for verification applications-the benefit of which includes the capability to verify an object without resorting to comparing with a reference original-it does not make sense to have a "private" fragile watermarking scheme.

3D features for watermarking

Feature-based watermarking schemes that embed a watermark W = $\{w_1, w_2, \bullet \; ..\}$ into a set of derived features F(0) = $\{f_1(O), f_2(O), \cdots\}$ represent one common approach towards watermarking an object. Usually, the feature set $\{f_1(O), f_2(O), \cdots\}$ is chosen such that slight modification of individual features does not perceptually degrade the functionality of the object, 0. In the image domain, it is the perceptual quality of an image. An example of such a set of features would be transform domain (like DCT and wavelet) coefficients, which contain significant energy content.

In a 3D model, the modification will have to be done on a set of features that will not significantly change the model's end uses and degrade the perceptual quality of the graphics generated from the model. A 3D object model may contain many types of information, including geometry, color of the vertices, directions of normals, and images for texture mapping.

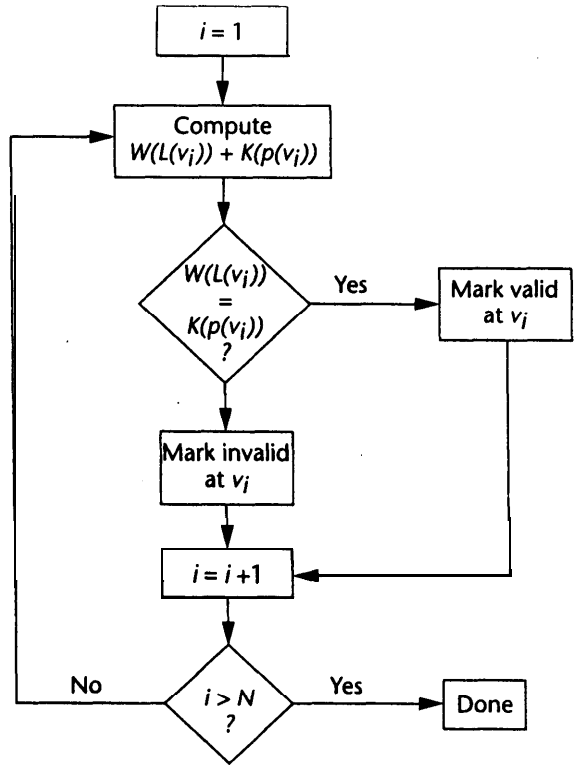
We will assume, without loss of generality, that the 3D objects of interest are described using a sequence of polygonal surfaces. In particular, each surface S will be of the form {vi, $v_2, \ldots, v_n$}, where v_i is a vertex in 3D space specified by its 3D coordinate (x_i, y_i, z_i). In describing our watermarking algorithm for verification, we will use only the surface information of an object (the sequence of surfaces $S_1, S_2, \cdots$), each of which is a polygon. In VRML,[9] surface information of an object is described using the `IndexedFaceSet` construct of the following form:

```
IndexedFaceSet {
      coord Coordinate {
            vertex [...]
      }
      coordIndex [...]
}
```

Polygonal surfaces are fundamental descriptors of 3D objects. While other descriptors exist, such as global lighting information, color of the vertices, directions of normals, and texture, they're not always necessary in describing 3D objects. In addition, it is straightforward to include them into the watermarking techniques, if desired. For example, in texture maps of images, 2D image watermarking techniques can be directly applied onto the images in addition to water-marking the polygonal 3D objects.

Watermarking 3D objects for verification

The technique we propose for watermarking 3D objects for image verification inherits several characteristics from the 2D image verification technique described by Yeung and Mintzer.[7] Common to both techniques are



2 A flowchart of the watermark decoding process.

- the processes of stamping an invisible watermark,
- decoding based on a verification key, and
- encoding based on making slight modifications to the object or image to meet certain criteria.

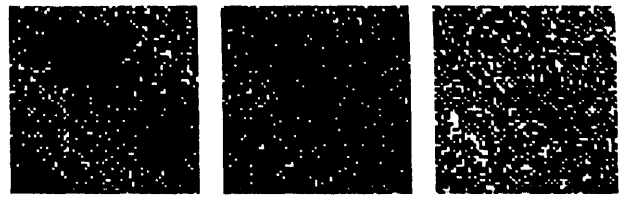
A key difference is that images lie on a fixed and regular grid, while 3D objects do not. Our technique overcomes this additional constraint (or rather, the freedom) inherent in 3D objects.

Before describing the watermarking technique, we need to state some notations. The list of 3D vertices is ordered as $v_1, v_2, v_3, \ldots$. The set of vertices, v_j's, are connected on some surface to v_i (these vertices are called the adjacent neighbors of v_i) such that $j < i$ and the vertex v_i itself is denoted by $N(v_i)$. The number of this collection is denoted by $|N(v_i)|$. A watermark signal, *W*, represents an ordered set of binary value used for insertion into the 3D object and for testing the authenticity of the objects. For the discussion that follows, we will assume *W* to be a 2D order set indexed by $W(i, j)$. A verification key, *K*, represents a set of lookup tables (LUT's) with binary entries and is necessary for decoding a 3D object.

We will describe the water-marking technique on one 3D object, 0. The generalization to several objects is straightforward. Following the notations introduced earlier, the watermarked object is denoted by $\hat{O}$, and 0' denotes a test object whose authenticity we want to verify.

Decoding the watermarked object

The goal of decoding is to extract a watermark, W, from the test object, 0', and compare against the watermark, *W*, to test for changes made to the 3D object, 0. Our scheme is a public scheme, meaning our decoding does not use the original object. We will describe *P*() (and thus the comparator C_δ of Eauation 1) later.



3 Three sets of location indices on 64 x 64 grids for three different objects: (a) a chair model with 450 vertices (shown in Figure 9), (b) a coffee pot model with 524 vertices, and (c) a tank model with 1,512 vertices (shown in Figure 8).

The process follows: Given a vertex v of the object 0', we first generate two values, the location index, $L(v)$, and the value index, p(v). *L(v)* is used to index into the watermark, *W*, for extraction of the bit value, *W(L)*. *(We* will refer to the indices as *L* and p when there is no ambiguity regarding the vertex of interest.) The value index, p, extracts the bit from the verification key, K, at location p, represented by $K(p)$. If the two bit values *W(L)* and $K(p)$ do not match, we say that the vertex v is invalid. The purpose of watermark embedding is to make sure that every vertex in a 3D object is valid according to the verification key, K, and watermark, *W*. We illustrate this process in Figure 2.

If an arbitrary key is used instead of the correct key, K, about 50 percent of the vertices will be invalid. Thus, without a'correct key, you cannot extract the watermark correctly.

Computing the location index. To generate the location index, *L*, we first compute N(v) of the current vertex v. The 3D coordinates of the vertices in $N(v)$ are then combined and hashed to produce the index *L*. The basic key step to generate a 2D index $L = (L_x, L_y)$ follows:

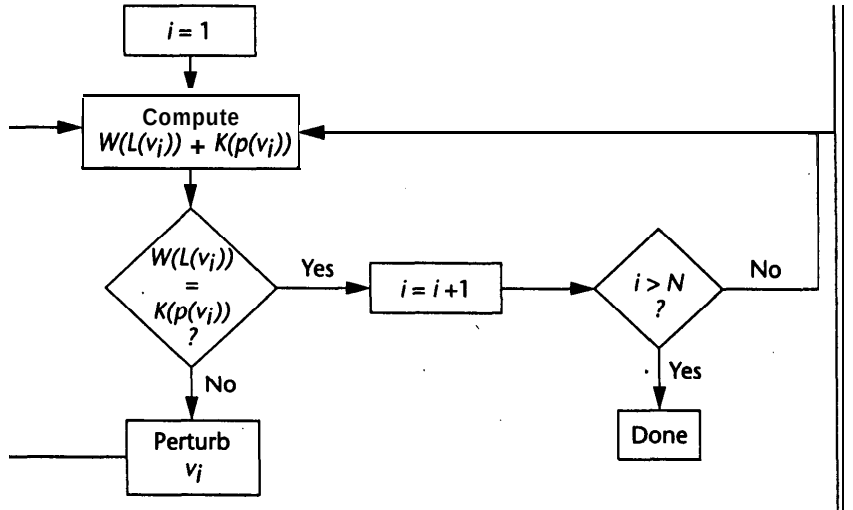
[Algorithm Compute Location Index]

1. Set $s = (s_x, s_y, s_z) = \frac{1}{|N(v)|} \Sigma_{u \in N(v)} u$
2. N_x = ConvertToInteger(s_x), N_y = ConvertToInteger(s_y), N_z = ConvertToInteger(s_z)
3. $L_x = f_x(N_x, N_y, N_z)$ and $L_y = f_y(N_x, N_y, N_z)$

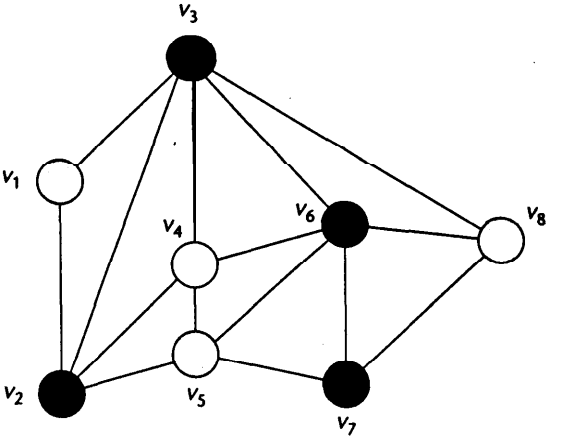
In step 1, we find the centroid of the vertices in N(v). This step seeks to combine the list of 3D coordinates into a three-tuple. Other methods, like the median and mode, can also be employed. Step 2 converts the floating-point numbers from step 1 into integers through the ConvertToInteger() routine. Step 3 then combines the three integers derived in step 2 to obtain two integers, L_x and L_y. An example f_x is (Nx + Ny + Nz) mod XSIZE. In summary, the three steps provide a mapping from the centroid (or other combinations) of the vertices in N(v) to a discrete 2D grid of size XSIZE by YSIZE. The location index, *L*, denotes the location on the grid. Finure 3 shows three sets of location indices corre-

sponding to three different objects. The 2D grid is chosen to be 64 x 64, and a white vertex represents an occupied 2D location.

Computing the value index. Next, to generate the value index, we take the 3D coordinate of v and map it into a three-tuple of integers, denoted by $p = (p_1, p_2, p_3)$. Any functions for scaling a floating-point rep resentation to integers (such as ConvertToInteger()) serves for this purpose. The value p is analogous to the color information of a pixel value. Thus, the combination of location L andvaluep are analogous to the combination of pixel location and color information in 2D images. Note, however, that mapping the collection of location indices does not cover all the 2D pixels in a regular grid.



4 A flowchart of the watermark embedding process.



5 An illustration showing an ordered traversal of the vertices.

The value index p is then used as the input into the key, K, to derive the bit $K(p)$. This bit value should match the watermark bit at location L-the bit $W(L)$ would validate the 3D vertexv, for example. The LUT in the key K can be generated using a pseudo-random number generator. The size of the LUT should also be sufficiently large to be secure. In our experiments, we use a key with 256 x 256 x 256 = 2^{24} entries. In fact, our LUT consists of three separate LUTs, K_1, K_2, and K_3, each with 256 binary entries. The value of $K(p)$ is then computed as

$$K(p) = K_1(p_1) \oplus K_2(p_2) \oplus K_3(p_3) \quad (2)$$

where $\oplus$ denotes binary XOR. We only need to maintain 256 x 3 = 768 binary entries.

Verification. One approach to compare the extracted watermark W and W is to count the number of mismatches between $K(p(v))$ and $W(L(v))$. Precisely, we can compute a number c, the correlation between two watermarks, as follows:

$$c = \frac{\left|\{v : K(p(v)) = W(L(v))\}\right|}{\|V\|} \quad (3)$$

When no changes have been made to 0, c = 1. To detect that no changes have been made to 0, we require that $\delta = 1$ in Equation 1.

All the detected invalid vertices in a 3D model can be highlighted (for example, using different colors at these vertices) at rendering time for quick identification and localization of the modifications through visual inspection. In general, the set of invalid vertices consists of modified vertices in a 3D object and some of their adjacent neighbors. The adjacent neighbors of a modified v may be flagged "invalid" because their location indices could be changed by the modification made to v. We will discuss some experimental results of such later.

Visual inspection of 3D models with a naked eye can be a time-consuming process and require intensive user interaction (rotating the model and zooming in on details to see changes). The difficulty of verification can grow in complex models. Another method allows the user to visualize the extent of the modifications made to object 0 using a 2D technique. This technique allows a user to immediately see the extent of modification (none, some, or a lot) and can be used with the 3D visual-inspection technique. We describe a technique to visualize and verify models in a 2D image plane later.

Embedding the watermark

The task of watermark embedding is to make slight changes to the geometry of an object such that

$$K(p(v)) = W(L(v)) \quad (4)$$

holds for every vertex v in the object. We will describe a technique for achieving this through the modification of only v. Generalization to include N(v) and other relevant information (such as lighting) is straightforward. We illustrate this embedding process in Figure 4.

An important consideration in computing Equation 4 is that of causality. Since we are perturbing v_i, it is essential to perturb them in the correct order ($v_1, v_2, \cdots$, for example). This causality concern is already built into the definition of $N(v_i)$, where the dependence is never on those unvisited vertices.

Figure 5 shows an ordered traversal and processing

(a) (b)

6 A 2D watermark of size 100 x 63 is a bitmap of a logo (a). The set of position indices for the tank model (b).

(a) (b) (c)

7 Visualizing the extracted watermarks: (a) extracted from watermarked object 0' , (b) extracted from a slightly modified version of watermarked object 0' , and (c)alteration detected (highlighted difference).

of some vertices. If the current vertex in processing is v_4, then vi, v_2, and v_3 are already processed. Then, the calculation of the location index $L(v_4)$ involves only the vertices v_2, v_3, and v_4.

To perturb a vertex v = (x, y, z) to v' = (x', y', z'), we first move the individual coordinate by Δ_x (or Δ_y or A,), and slowly increment the perturbation until Equation 4 holds. The first few steps are

$$
\begin{aligned}
&1.\ v' \leftarrow (x + \Delta_x, y, z),\\
&2.\ v' \leftarrow (x - \Delta_x, y, z),\\
&3.\ v' \leftarrow (x, y + \Delta_y, z),\\
&4.\ v' \leftarrow (x, y - \Delta_y, z),\\
&5.\ v' \leftarrow (x, y, z + \Delta_z),\\
&6.\ v' \leftarrow (x, y, z - \Delta_z),\\
&7.\ v' \leftarrow (x + \Delta_x, y + \Delta_y, z),
\end{aligned}
$$

and so on.

Visualizing the decoded watermark

After a watermark has been extracted from a 3D object, the number of times Equation 4 is not satisfied can indicate the amount of modification made to the object. The invalid vertices can be highlighted in 3D using rendering time for visual inspection. Another method allows the user to visualize the changes in 2D. This eliminates the need for constant user's interaction with the system (no zooming, panning, and rotating of the 3D model). To achieve this, we select a 2D-watermarksignal, W, which is meaningful whenviewed as an image. Figure 6a shows an example watermark, W, which is a bitmap of a logo. Figure 6b shows the set of position indices of a 3D object on the same 2D grid, upon which W is based.

To visualize the extent of modification made to a 3D model, we propose to superimpose the location indices onto the watermark signal W. We construct a grayscale image W' from W, the set of location indices of 0', $L(O')$, the sets $G_1 = \{v \in 0' : K@(v)) = W(L(v))\}$, and $G_2 = \{v \in 0' : K@(v)) \neq W(L(v))\}$. The image W' consists of only 4 values: 0, 64, 192, and 255. To begin the process, we set the pixels in W'' to take on the value 192 if the corresponding bit (at the same location) of W is 1, and 64 if the bit is 0. Then, on the locations specified by $L(O')$ we modify the pixel values to either 0 or 255, according to whether the corresponding v is in G_1 or G_2. For those valid v's in G_1, the pixels with value 64 become 0, and pixels with value 192 become 255. On the other hand, for those invalid v's in G_2, the pixels with value 64 become 255, and pixels with value 192 become 0. The effect is to both highlight the location indices (using pixel values 0 and 255) and to indicate those locations where the invalid v's were mapped. We call the process visualizing the extracted watermark.

If we perform a watermark extraction process for unwatermarked objects, statistically about 50 percent of the extracted pixel value v's are invalid. This acts like superimposing a noise pattern onto the watermark image W: the processes of changing the pixel values from 192 to 0 or from 64 to 255 significantly degrade or even destroy the watermark image for visualization purposes. On the other hand, the extracted results from a watermarked object clearly show the watermark image. Figure 7 shows W' of the object after watermark extraction and the results after modification of the watermarked object. Figure 7a shows W" of a watermarked object that has not been modified. Here, we can clearly see the logo image. Further-mare, pixels with value 0 and 255 indicate the position indices. Finally, Figure 7b shows W'' of a watermarked object that has been slightly modified (a view of the object is shown in Figure 8c). We use a red circle on a location indexhighlighting modified 3D vertices in Figure 7c. Comparing 7a and 7b, we can see that a few pixels differ. These locations represent the location indices of those 3D vertices that have been modified.

Some enhancement features

A few features will help enhance the watermarking process. Specifically, we will address those discussed earlier.

Detecting object cropping. Protection against cropping-rather, detection of object cropping attempts-results from the dependencies built in when the location index is computed: when a neighboring vertex u of a vertex v is removed, a different L results. However, in an alternative to the scheme, further protection via cropping detection may also be added by making the computation of the position index, p(v), also dependent on the neighbors of vertexv. For the fol-

lowing discussion, we will denote $N(v_i) = \{u_1, u_2, \ldots, u_{Mi}\}$, and $p(v_i) = \{p_1(v_i), p_2(v_i), p_3(v_i)\}$. $K(p(v_i))$ can instead be computed as

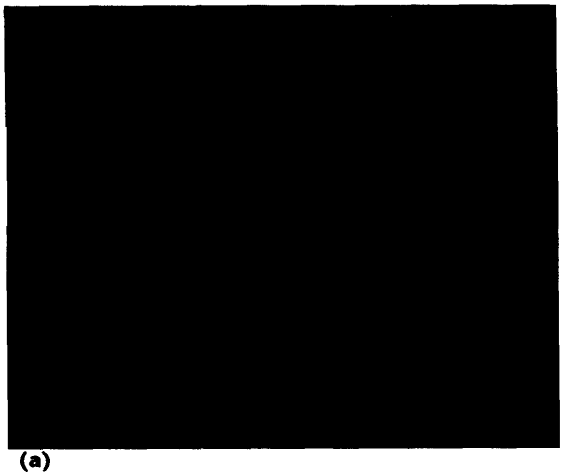
$$K(p(v_i)) = K_1(p_1(u_1)) \oplus K_2(p_2(u_1)) \oplus K_3(p_3(u_1)) \oplus K_1(p_1(u_2)) \oplus K_2(p_2(u_2)) \oplus K_2(p_2(u_2)) \oplus \ldots \oplus K_1(p_1(u_{Mi})) \oplus K_2(p_2(u_{Mi})) \oplus K_3(p_3(u_{Mi})) \quad (5)$$

where $\oplus$ denotes binary XOR. Thus, the lookup table K is applied not just to the position indices of the current vertex, v_i, but also to all of its other elements of $N(v_i)$. When some of the neighbors are missing, due to cropping for example, a possibly different $K(p(v_i))$ results. In other words, cropping will produce a mismatch on about half of the extracted results on the boundaries.

Guarding against potential attacks. In accessing the scheme's resistance against tampering, we have to look at how easy it is to forge a "good" watermark after an alteration without the knowledge of the key (or the set of binary LUTs). In other words, the alteration will go undetected in the verification process. Deciphering the content of the set of tables proves tedious thanks to the key length (768 bits) and the dependencies of one vertex on its neighbors in the decoding of the watermark, which make the inversion computationally infeasible without knowledge of the watermark. The exhaustive inversion computation may be made slightly easier by having complete or partial knowledge of the watermark image. For example, while it is visually pleasant to have corporate logos as watermark images, such a practice can lead to an attacker guessing or experimenting to find partial information on the watermark for subsequent attacks. Though not often feasible, we can guard against potentially sophisticated attacks by further scrambling the watermark with a key. This key can be the same or different from the one used to generate the watermark extraction function. Alternatively, we could modulate the watermark image into some random-looking patterns using some noise signal, which in turn can serve as the key in decoding the "chaotic image mixing" technique. In this case we need 'to incorporate an additional step of descrambling the watermark prior to visual verification.

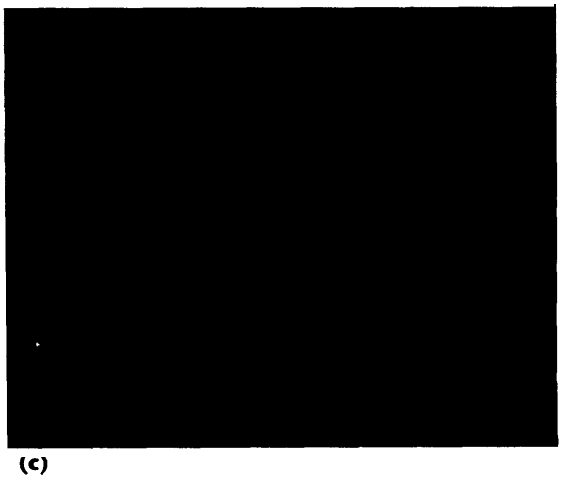
Results and discussion

Figure 8 shows a tank model at different stages of watermarking. Figure 8a shows the original model 0. Figure 8b shows the same of view of the watermarked model $\hat{O}$. We chose this view to show as many 3D vertices as possible and to illustrate that even after watermarking, the two models are visually identical. In Figure 8c, we make a slight change to $\hat{O}$ to form 0': the top of the tank is now made sharp and pointed. To make this change, the locations of 12 vertices are changed in $\hat{O}$ to form 0'. The decoding reports that 15 vertices are invalid. Of the 15 vertices, 12 vertices have been modified, with the remaining 3 vertices classified as invalid because some of the modified 12 vertices changed the location indices of neighboring vertices. This violates the right-hand side of Equation 4, even though the left-hand side does not change. Figure 7c lets us visualize the extracted watermark from this modified object in 2D. From this view, we can immediately see that only slight modifications have been made to the 3D object.



(a)

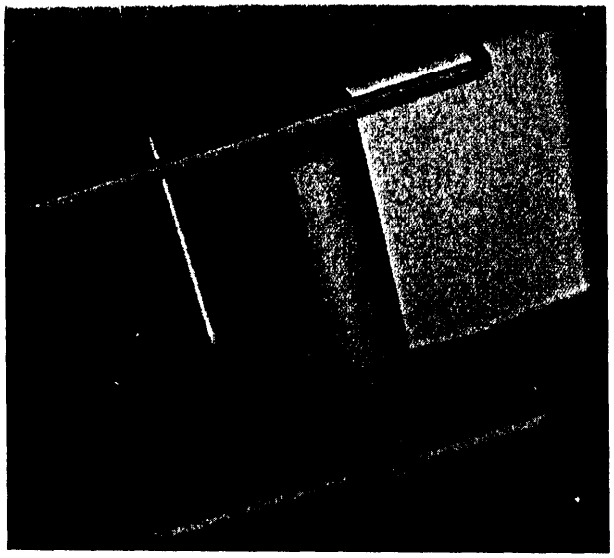
(b)



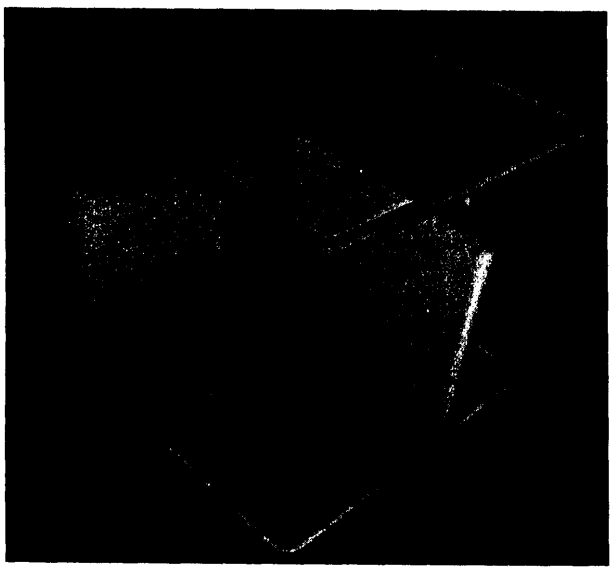
(c)

8 A tank model showing (a) a view of the original, (b) the same view after the original has been watermarked, and (c) the same view after the watermarked original has been modified.

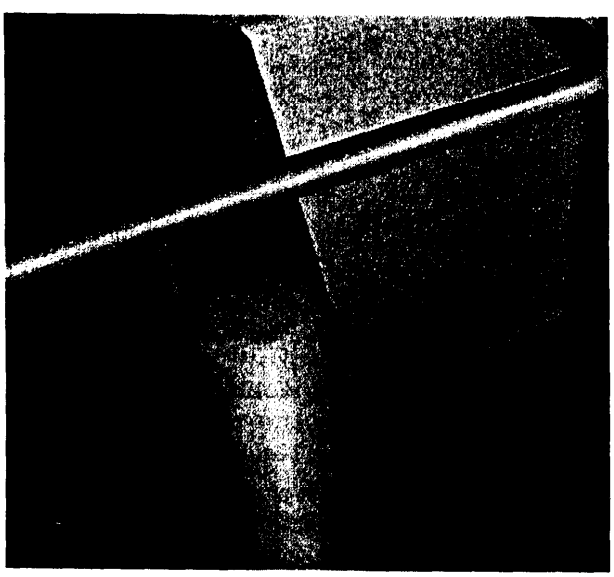
Generally, decoding will report more vertices as invalid because of using adjacency information. However, adjacency information is crucial in describing 3D objects-and thus cannot be omitted. We can further study methods that precisely locate the altered vertices and polygons. We will also look into the techniques tha



(a)



(b)



(c)

9 A watermarked chair model: (a) one view, (b) another view, and (c) a zoomed-in view.

incorporate explicit authentication and verification of partial VRML models. Currently, verification of a partial model is implied: all vertices and polygons in the partial model can be verified except those on the outermost boundaries, which will appear to have been altered since watermarking.

In Figure 9, we show a watermarked version of a simple chair model. This example is chosen to illustrate that even on models with simple surfaces, watermarking does not induce visible artifacts. Figures 9a and 9b show different views of the chair. Figure 9c shows a zoomed-in view. Even in the zoomed-in view we cannot perceive any artifact due to watermarking.

For a fragile watermarking scheme to be effective and secure in verification and authentication applications, it must be very difficult-better yet impossible-for an interloper to decide whether an object has been watermarked, what the watermark is, and where the information is embedded. This must be done in such a manner that an attacker cannot reapply the correct watermark after an alteration to fool the verification process. In this sense, the experimental results obtained indicate that our verification achieves this goal. Without the proper key, it's very unlikely that anyone could extract the watermark, or even decide if the model is a watermarked version. In addition, an interloper, after altering the model, cannot remap the geometry to restore the changed watermark data to its original version without knowing the key and the watermark.

Our proposed technique allows us to detect, localize, and visualize the locations and extent of modifications made to the geometry of 3D objects. To extend the idea to indicate modifications to other relevant information, such as color, normals, and texture, we can incorporate these parameters, where available, into the mapping and hashing processes in computing location and value indices. This allows a multiple-level verification in which, at the most fundamental level, only the geometry is verified, then the lighting information, followed by texture mapping. Color and texture maps can be watermarked also by image-based techniques for respective verification. The incorporation and integration of these other features warrant future investigations.

Conclusions and future work

In this article, we have presented definitions and formulations for generalized watermarking of 3D objects. We also proposed new watermarking techniques for 3D object verification to detect unauthorized alterations in the underlying polygonal models. The goal of such watermarking techniques is to embed watermarks that are sensitive to slight modification and to allow quick detection, localization, and visualization of the modifications. We achieved verification and authentication without resorting to the original model for comparison.

We have discussed some future investigations, which include developing visualization tools to detect and localize alterations, and the techniques that can provide 'enhanced functionality like authentication and verification of partial VRML models and multilevel authentication of 3D models.

Fragile watermarking is only one category of watermarking. To broaden the field of applications, we also need to study robust watermarking of 3D objects. Dif-

ferent sets of criteria and challenges exist for robust watermarking.

Watermarking technologies are new to many researchers in the signal (audio/image/video) processing and steganography fields, and even newer to the computer graphics community. Inherently, digital watermarking of objects, like its image counterparts, proves a difficult problem. Understandably, the existing technologies, in particular robust watermarking techniques, may not yet have achieved their ambitious goals and lived up to the high expectations. Nevertheless, the topic is an important one in any field that involves media content creation, manipulation, and delivery—computer graphics not excluded. We hope introducing and investigating the topic will help generate more discussions and prompt future work in this area. ■

References

1. R.Ohbuchi, H. Masuda, and M. Aono, "Watermarking Three-Dimensional Polygonal Models through Geometric and Topological Modifications," *IEEE J. on Selected Areas* in Comm., Vol. 16, No. 4, May 1998, pp. 551-560.
2. W.R. Bender, D. Gruhl, and N. Morimoto, "Techniques for Data Hiding," *Proc. SPIE: Storage and Retrieval of Image and Video Databases,* Vol. 2420, 1995, pp. 164-173.
3. I.J. *Cox* et al., *Secure Spread Spectrum Watermarking for Multimedia,* Tech. Report TR-95-10, NEC Research Institute, Princeton, N.J., 1995.
4. N. Nikolaidis and I. Pitas, "Copyright Protection of Images using Robust Digital Signatures," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* IEEE Press, Piscataway, N.J., 1996.
5. R.G. van Schyndel, A.Z Tirkel, and C.F. Osborne, "A Digital Watermark," *Proc. IEEE Int'l Con, Image Processing,* Vol. 2, IEEE Press, Piscataway, N.J., 1994, pp. 86-90.
6. M. Swanson, B. Zhu, and A. Tewfik, "Transparent Robust Image Watermarking," *Proc. Int'l Con. Image Processing* 96, Vol. 3, IEEE Press, Piscataway, N.J., 1996, pp. 211-214.
7. M.M. Yeung and EC. Mintzer, "An Invisible Watermarking Technique for Image Verification," *Proc. Int'l Conf. Image Processing* 97, Vol. 2, IEEE Press, Piscataway, N.J., 1997, pp. 680-683.
8. J. Zhao and E. Koch, "Embedding Robust Labels into Images for Copyright Protection," *Intellectual Property Rights and New Technologies, Proc. KnowRight Conf.,* 1995, pp. 241-251.
9. ISO/IEC 14772-1:1997, *The Virtual Reality Modeling Language,* 1997, http://www.vrml.org/Specifications/VRML97/.
10. S. Craver, B.L. Yeo, and M.M. Yeung, "Digital Watermarking-Technical Trials and Legal Tribulations," *Comm. ACM,* Vol. 41, No. 7, 1998, pp. 45-54.

Boon-Lock *Yeo manages the Video Technology department for the Microcomputer Research Labs of* Intel Corporation. He received his *BSEE from Purdue University in 1992, and his MA and PhD degrees from Princeton University in 1994 and 1996, respectively. He holds four US patents, has 73* pendingpatent applications, *and was the recipient* of the *1996 IEEE Circuits and Systems Society Video Technology Transactions Best Paper Award. He is an Associate Editor for* IEEE Transactions on Image Processing. *His interests are in the general areas* of *image and video processing.*

Minerva M. Yeung *is currently a Senior Staff Researcher and manager* of *a research group working on media protection and management* for the *Microcomputer Research Labs of Intel Corporation. She received a BSEE (with Highest Distinction) from Purdue University in 1992, and her PhD and MA degrees from Princeton University in 1996 and 1994,* respectively. She is an associate editor of *the new* IEEE Transactions on MultiMedia, *a guest editor* of *a special issue* of *the* Communications of the ACM *on digital watermarking, and a co-chair* of *the Storage and Retrieval* of *Images and Video Databases conference in 1999. Her research interests are in the general areas* of *image and video processing, content protection, computer-human interaction, and multimedia information systems.*

Contact Yeo at Microcomputer Research Labs, Intel Corp., SC12-303, 2200 Mission College Blvd., Santa Clara, CA 95052-8119, e-mail boon-lock.yeo@intel.com.